Malaysian Journal of Mathematical Sciences 10(2): 145–178 (2016)

MALAYSIAN JOURNAL OF MATHEMATICAL SCIENCES
Journal homepage: http://einspem.upm.edu.my/journal

Hybrid Genetic Algorithm for University Examination Timetabling Problem

S. Ishak[1], L. S. Lee *[1,2], and G. Ibragimov[1]

[1] *Department of Mathematics, Faculty of Science, Universiti Putra Malaysia, 43400 Serdang, Selangor, MALAYSIA.*
[2] *Laboratory of Computational Statistics and Operations Research, Institute for Mathematical Research, Universiti Putra Malaysia, 43400 Serdang, Selangor, MALAYSIA.*

E-mail: lls@upm.edu.my
**Corresponding author*

ABSTRACT

This paper considers a Hybrid Genetic Algorithm (HGA) for University Examination Timetabling Problem (UETP). UETP is defined as the assignment of a given number of exams and their candidates to a number of available timeslots while satisfying a given set of constraints. Solutions for uncapacitated UETP are presented where five domain-specific knowledge in the form of low-level heuristics are used to guide the construction of the timetable in the initial population. The main components of the genetic operators in a GA will be tested and the best combination of the genetic operators will be adopted to construct a Pure Genetic Algorithm (PGA). The PGA will then hybridised with three new local optimisation techniques, which will make up the HGA; to improve the solutions found. These new local optimisation techniques will arrange the timeslots and exams using new explicit equations, if and only if, the modification will reduce the penalty cost function. The performance of the proposed HGA is compared with other metaheuristics from literature using the Carter's benchmark dataset which comprises of real-world timetabling problem from various universities. The computational results show that the proposed HGA outperformed some of the metaheuristic approaches and is comparable to most of the well-known metaheuristic approaches.

Keywords: university examination timetabling problem, local optimization techniques, genetic algorithm, hybrid, uncapacitated.

1. Introduction

In the last few decades, the timetabling problem has received special attention from the operations research community. Timetabling concerned in assigning objects (e.g. people, vehicles, machines or exams) subject to a set of constraints in a pattern of time or space. There are many examples of scheduling and timetabling such as scheduling of employees' shifts and working hours, transit route for urban transit scheduling problem, constructing timetables for exams and courses in educational institutions, scheduling of sports or business events, etc. The University Examination Timetabling Problem (UETP) is a classic combinatorial optimisation problem which has to be tackled by universities from all over the world. This problem shares a common framework which is to allocate exams to conflict-free exam timeslots (i.e. no student has to sit for more than one exam simultaneously).

Nowadays, many universities are introducing the concept of cross-faculty. This will allow the students to have much greater flexibility in enrolling courses that they want to take as well as giving a much greater choice to them. The exams timetable will be more difficult to construct as the number of enrolment at the university increases every year. The administration department of the university needs to solve the timetabling problem in order to plan their courses or exams by allocating these events to specific timeslots, rooms and lecturers.

The investigation conducted by Burke et al. (1996a) found that the constraints in the domain of UETP may vary from one university to another. The hard constraints' costs must be satisfied under all circumstances for the timetable to be feasible whereas for the soft constraints, these are desirable to satisfy but not absolutely important. Due to the large variety of the problem presented, the common hard constraints cost to be considered are as follows:

1. certain exams need to assign to specific timeslots or can only be held in a limited set of timeslots.
2. certain exams must satisfy pair wise schedule.
3. certain exams may require specific rooms.
4. there may be restrictions on students' individual exam timetables.

5. large exams should be scheduled earlier in the timetable.
6. student must not sit for more than x exams in any y consecutive timeslots.

UETP can be categorized as either capacitated or uncapacitated problem. In the capacitated UETP, rooms to allocate the exams are taken into consideration and the number of students sitting for the exams must not exceed the available seat of the rooms, while in the uncapacitated UETP, room capacities are not considered. In this paper, we concentrate on finding the solutions for uncapacitated UETP.

The main goal of this paper is to design a Hybrid Genetic Algorithm (HGA) for solving uncapacitated UETP. We investigate how a Pure Genetic Algorithm (PGA) when hybridised with three new local optimisation techniques can improve the overall quality of the solution. The first technique focuses on inserting a scheduled exam to a new timeslot. Second technique is concerned with swapping of two scheduled exams between two different timeslots while the third technique deals with interchanging the timeslots in the timetable. These new local optimisation techniques will arrange the timeslots and exams using new explicit equation, if and only if, the modification will reduce the penalty cost function. The remainder of this paper is organised as follows: Section 2 explores the solution approaches in solving the UETP. In Section 3, the PGA where each main component used in a GA is explained. Section 4 discusses the HGA used in this paper where three new local optimisation techniques are discussed in detail. The computational results and discussions are presented in Section 5. Finally, the conclusion is given in Section 6.

2. Solution Approaches

Early approaches have been widely applicable to solve the UETP since 1960's using the concept of heuristic ordering (Carter et al., 1996). These methods firstly order exams using heuristics knowledge and then scheduled the exams sequentially into feasible timeslots so that no exam in the timeslots will clash with each other (Carter, 1986). Table 1 shows several sequencing strategies listed by Carter et al. (1996).

Table 1: Sequencing Strategies by Carter et al. (1996)

Heuristics	Descriptions
Random Ordering (RO)	Exams to be scheduled are chosen at random.
Largest Degree (LD)	Exams are scheduled according to the number of conflict the exams have with other exams. Exams with the highest number of conflicts will be scheduled first.
Largest Enrolment (LE)	Exams are scheduled according to the number of students enrolled for the exams. Exams with the highest number of students enrolled will be scheduled first.
Largest Weighted Degree (LWD)	Similar to LD, but exams are weighted according to the number of students involved. Exams with the highest number of students conflict will be scheduled first.
Saturation Degree (SD)	The next exams to be scheduled are based on the number of available timeslots for the exams at the current time. Exams with the least number of available timeslots will be scheduled first.

Burke et al. (1996b) studied a Memetic Algorithm for UETP where each solution in the population is represented as a number of memes that contained information on which exams are scheduled and where the exams will be carried out for a particular timeslot. A weighted roulette wheel is used to construct the initial population to choose which timeslot to schedule for each exam. Light and heavy mutation operators followed by hill climbing operator are employed in the initial phase in order to find fast and good solutions. Another contribution of this paper is the introduction of a new benchmark dataset of the UETP namely the Nottingham dataset which involved 805 exams with 7,896 students.

Reis and Oliveira (1999) proposed a constraint logic programming language ECLiPSe to a large UETP of the University of Fernando Pessoa in Porto with more than 3500 students, from 21 different courses correspond to 314 exams. Four main labelling strategies are employed. In the first strategy, the algorithm dealt with the timeslot variables, then the room variables and finally with invigilator variables. For second strategy, they started with room variables, followed by timeslot variables and ended with invigilators variables. The third strategy started with timeslot variable, continued simultaneously to room and invigilator variable labelling while for the last strategy, all three types of variables were dealt simultaneously. For typical UETP, the best strategies are the first and the third strategies. The results obtained by the algorithms showed three times better than the manual solutions.

In the same year, Chu and Fang (1999) compared the performance of a GA and a Tabu Search (TS) to allocate 10 exams into 12 timeslots (in 3 days) which involved 50 students. In the GA, they used the same representation as reported in Corne et al. (1994) where each timetable represented as an array and the length of the chromosome represented as a list of number of exams to be scheduled. They employed one point crossover by swapping the timeslots. For the TS, the same representation is used as in the GA where 20 constructed timetables are produced and sorted according to their fitness value. The solutions for the next iteration are produced by choosing the best solution and randomly swapped two exams timeslots. Best solution found in this step will be selected to test with two basic mechanisms in their TS: tabu restrictions and aspiration criteria. The results showed that the TS produced better solution compared to the GA. However, the GA approach managed to produce several near optimal solutions.

Casey and Thompson (2003) proposed a two-phased approach in GRASP technique to solve the UETP. In the first phase, initial solution is generated using the greedy approach achieved by ordering the examinations according to one of the low-level heuristics (LD, LWD, LE and SD). Roulette wheel selection is used to choose the top exams in the list for the next exam to be scheduled. A backtracking with tabu list is employed to eliminate the clashes. Tabu list is used to forbid cycling during the search process. In the second phase, Simulated Annealing (SA) is used with high starting temperature and fast cooling to minimize the soft constraint costs. The proposed algorithm is applied to the benchmark datasets and the results obtained show that the SD heuristic employed in the first phase is able to produce the best timetable compared to other low-level heuristics.

Burke and Newall (2004) presented two variants of local search algorithms: a time-predefined variant of SA and a basic variant of linear decreasing of the upper boundary value (ceiling degrading) where the algorithm accepts the worse candidate solutions during the execution as its fitness is less than or equal to the given upper boundary value. The basic geometric cooling algorithm is employed to make SA run for a definite number of moves and terminate until reaching a temperature. Comprehensive experiments are carried out to analyze the trade-off between the time and solution quality on problems of different size. The results obtained showed that the approaches significantly outperform when compared to the previous best result on the Carter's and Nottingham's benchmark dataset problems.

Kendall and Hussin (2005) developed a TS based hyper-heuristic framework to solve UETP where the hyper-heuristic module is used to design and test the strategies that help in making an intelligent decision and guide the search to either improve or diversify the exploration of the search space. Constructive heuristic is used to construct the initial solution followed by a randomization of heuristics to explore the neighbourhood for a better solution. Tabu list is used to store information about heuristics while tabu duration decide how long a heuristic should remain tabu. The algorithms are tested on Carter's benchmark dataset. Even though the results obtained are not able to beat the best result amongst the literature but their approach able to produce a quality solutions.

In the same year, Asmuni et al. (2005) made some modifications to the sequential construction algorithm applied by Carter et al. (1996). They applied the combination of two ordering heuristics out of three ordering heuristics (LD, LE and SD) simultaneously to all the exams and find the clash free timeslots with least penalty cost to schedule all exams into the timetable. In the case where several timeslots is proposed, the exam will be scheduled to the last available timeslot in the list. Fuzzy model are serve to represent the knowledge that are imprecise, uncertain, or unreliable where the fuzzy functions are used to give the evaluation for the ordered exams.

Abdullah and Burke (2006) developed a basic methodology of Ahuja-Orlin to construct both capacitated and uncapacitated UETP where the approach is based on an improved graph representation of solution adjacency. In this study, a very large neighbourhood structure is combined with the technique of identifying improvement. A modified shortest path label-correcting algorithm is used to identify the improvement move by finding cycle exchange operations. The computational experiments showed that when defining a neighbourhood structure, cyclic exchange operation is superior when compared to simply using two exchanges. In this approach, exams are divided into cells and each cell is assigned a timeslot. Their approach is tested on the Carter's benchmark problem and is evaluated against the other methodologies in the literature. The results obtained outperform some of the best known results.

In 2008, Asmuni (2008) presented fuzzy methodologies to solve both university course and exams timetabling problem. In the construction stage, fuzzy techniques are combined with multiple heuristic orderings to generate the timetable at the beginning. In the multiple heuristic ordering, different combinations are examined using two and three heuristic ordering simultaneously. The methods are tested on the Carter's benchmark datasets. Experimental results showed that the usage of fuzzy multiple heuristic orderings with parameter tuning obtained better results compared to the single heuristic orderings.

Pillay and Banzhaf (2008) proposed a developmental approach based on cell biology for the uncapacitated problem. A mature organism which is presented as the solution of the timetable problem is developed through the process of cell division, cell interaction and cell migration. Process of scheduling starts with the creation of a single cell and exams were scheduled according to SD. If there is no feasible timeslot to slot in the exams, cell division will occur where cell is divided into two daughter cells, one cell contained the clash exam and another cell contained the rest of the exams. In cell migration, a cell is moved from one region of organism to another. Two types of cell migration are studied: random migration and stimulus-driven migration. In each iteration, cell interaction occurred in order to decrease the soft constraints cost. Their approach is tested on Carter's benchmark dataset and the results obtained are comparable with those produced by other biologically inspired algorithm.

Abounacer et al. (2010) developed a hybrid Ant Colony Optimization (ACO) algorithm with the complete local search with memory (CLM) heuristic to resolve the UETP. CLM is a new neighborhood search approach that uses memory structures where all explored solutions are recorded in a list to prevent the exploration of solutions that have been already visited. The exams are sorted using LD and LE strategies. Their proposed algorithms produced comparable results using Carter's benchmark datasets.

Pillay and Banzhaf (2010) employed two phase approaches in solving UETP by using GA and proved that the effectiveness of using domain specific knowledge in the form of heuristics during the construction of timetable in the first phase. They also introduced new low-level heuristics namely highest cost (HC). They found that the combination of HC with SD heuristics and used the LWD to break ties when performing a Pareto comparison of exam has produced the best result. While in second phase, the GA refines the solutions found in the first phase over a set number of generations by using mutation operator with the aim of minimising the soft constraint cost of the solutions.

In 2012, Pillay (2012) studied the effect of different representations against the quality of the timetable. The evolutionary algorithm (EA) implemented with each of the representation. By using tournament selection and one point crossover, they evaluated three representations; fixed length heuristic combination (FHC), variable length heuristic combination (VHC) and N-time heuristic combination (NHC) with 50 runs each. The author found that VHC representation performed better than NHC and FHC representation. An EA- based hyper-heuristics which combined with all three representation namely CEA was tested and found that it performed better than EA- based hyper-heuristics using FHC, VHC and NHC.

Abdul-Rahman et al. (2014) investigated an adaptive decomposition strategy that grouped the exams into two sets: difficult and easy sets to solve the UETP. In the construction stage, difficult set are filled with the exams that cannot be scheduled and cause the infeasibility to the timetable while the examinations that feasible to be scheduled are assigned in the easy set. A new subset (called the boundary set) is introduced in the easy set to apply the shuffling strategies that shuffled the given ordering of exams. In the difficult set, the quality of the timetables could be improved by merging or swapping the boundary set. Roulette wheel selection is employed in order to shuffle the ordered exams. The proposed algorithm is tested on the Carter's benchmark problems and the results obtained are comparable to existing constructive approaches.

3. Pure Genetic Algorithm

In this section, series of experiments were carried out to find the best combination of each of the genetic operators in the proposed PGA for solving uncapacitated UETP. When using a PGA to solve an optimization problem, the most important part is to choose a suitable gene representation for the problem. The choice made is for the chromosome to be an ordered list of numbers. Figure 1 illustrates an example of the integer representation used for this problem. Given there are 10 timeslots ($t_i, i = 1, 2, \ldots, 10$) with 20 exams ($e_j, j = 1, 2, \ldots, 20$) to be scheduled. Exams e_3, e_5 and e_7 are assigned to timeslot t_1, exams e_1, e_4 and e_{18} are assigned in timeslots t_2, and exams e_{13} and e_{14} are assigned to timeslot t_{10}.

	Gene (Timeslot)									
	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
Chromosome: (Timetable)	e_3	e_1	e_2	e_{10}	e_9	e_8	e_{12}	e_{19}	e_{16}	e_{13}
	e_5	e_4	e_6		e_{17}	e_{11}	e_{15}		e_{20}	e_{14}
	e_7	e_{18}								

Figure 1: Gene Representation of PGA for Uncapacitated UETP

For initial population, two types of initialization methods which are the random initialization method and the low-level heuristic construction method are applied. In the first method, exams are randomly ordered before assigned to feasible timeslots. For the second method, domain specific knowledge in the form of heuristic is used to guide the construction of the timetable. The low-level heuristics used in this paper are LD, LWD, LE, SD and HC. By referring to the results of Pillay and Banzhaf (2010), they found that the combination of

SD and HC with the used of the LWD and LE heuristics to break ties produced the best results. Hence, in this paper, we propose to use 10% of the total exams to be scheduled with the combination of LD, LWD and LE, while the rest are the combination of SD and HC in order to focus on reducing the value of the penalty cost. In the case of ties, the exam which is appeared first in the list is given the priority to be scheduled first.

In a real world situation, it is impossible to fulfil all the soft constraints, but the quality of a timetable will be improved by minimizing these violations. In this paper, the soft constraint cost considered is the spread of the exams that have common students throughout the exam period so that the students may have enough time to do their revision. The following proximity cost function defined by Carter et al. (1996) is used to calculate the quality of the solutions for the soft constraint cost:

$$\mathbf{C} = \frac{\sum\limits_{i=1}^{M-1} \sum\limits_{j=i+1}^{M} \text{proximity}\left(t_{e_i}, t_{e_j}\right) \cdot \text{com}_{ij}}{N}, \tag{1}$$

where

$$\text{proximity}\left(t_{e_i}, t_{e_j}\right) = \begin{cases} 2^{(5-\Delta t)} & \text{if } 1 \leq |\Delta t| \leq 5 \\ 0 & \text{otherwise} \end{cases}. \tag{2}$$

Equation 1 represents the proximity cost function on the average penalty per student where M is the total number of exams, N is the total number of students and com_{ij} is the total number of common students between exams i and j. Equation 2 calculates the proximity value between two timeslots where exams i and j are scheduled in timeslots t_{e_i} and t_{e_j} respectively, and $\Delta t = |t_{e_i} - t_{e_j}|$,where $i \neq j$. For instance, a proximity value of 16 is assigned if a student has two exams consecutively (i.e. $\Delta t = 1$), the value of 8 is assigned if a student has two exams with a gap timeslot in between, and so forth. The aim of this study is to minimize the value of Equation 1. Note that a student is not allowed to sit for two exams simultaneously.

In this study, a computational comparison is made among the three standard selection mechanisms to select parents from the population. There are roulette wheel, rank and tournament selections. For crossover operator, three types of crossover operators are used namely one-point, two-point and position-based crossovers. However, some modifications are required for the above mentioned crossovers in order to produce feasible solutions. For the illustration purposes, Figure 2 below is used to represent the parents for these three crossover operators. Each parent consists of 10 exams scheduled in 5 timeslots.

Parent 1:

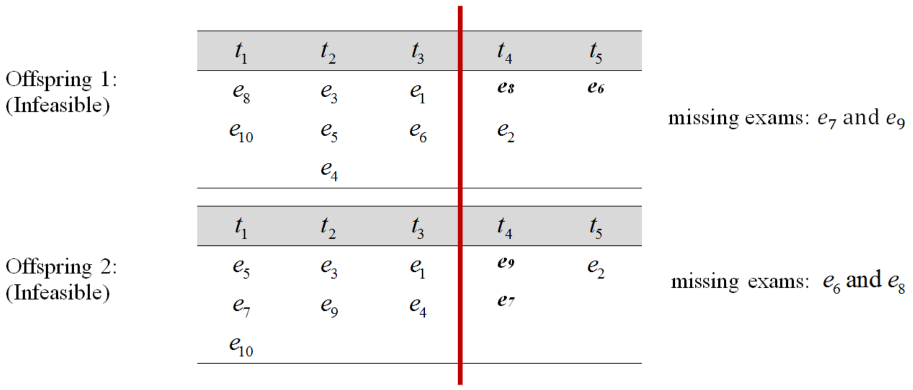
t_1	t_2	t_3	t_4	t_5
e_8	e_3	e_1	e_9	e_2
e_{10}	e_5	e_6	e_7	
	e_4			

Parent 2:

t_1	t_2	t_3	t_4	t_5
e_5	e_3	e_1	e_8	e_6
e_7	e_9	e_4	e_2	
e_{10}				

Figure 2: Selected Parents for Crossover

In the modified one point crossover (M1PC) (see Figure 3), a random crossover point is selected to be the crossover point between the parents. The symbol "|" indicates the randomly chosen crossover point. Then, the first part of the Parent 1 and 2 (before the crossover point) are copied into new Offspring 1 and 2. The remaining part of Offspring 1 and 2 are then filled with the second part of Parent 2 and 1 respectively. Figure 3 shows the offspring that are produced after the interchanging process, where the crossover point is between gene 3 and gene 4.



Offspring 1: (Infeasible)

t_1	t_2	t_3	t_4	t_5
e_8	e_3	e_1	e_8	e_6
e_{10}	e_5	e_6	e_2	
	e_4			

missing exams: e_7 and e_9

Offspring 2: (Infeasible)

t_1	t_2	t_3	t_4	t_5
e_5	e_3	e_1	e_9	e_2
e_7	e_9	e_4	e_7	
e_{10}				

missing exams: e_6 and e_8

Figure 3: Infeasible Offspring 1 and Offspring 2 of M1PC

Note that after interchanged the second part of the parents, both offspring do not represent feasible solutions, where all exams must be scheduled in the timetable (i.e. no missing exams) and each exam must only be scheduled once (i.e. no duplication of exams). For instance, in Offspring 1, duplication of exams have occurred in exams e_6 and e_8, while the missing exams are e_7 and

e_9. Therefore, for the second part of Offspring 1, the duplicated exams (e_6 and e_8) must be removed from the timetable and the missing exams (e_7 and e_9) must be rescheduled back in the timetable. The following steps are performed in order to produce feasible offspring for the M1PC:

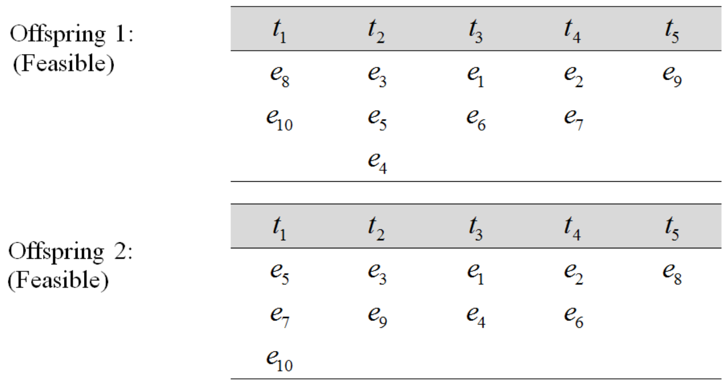
Step 1: Identify and remove the duplicated exams in the second part of Offspring 1.

Step 2: Reschedule the remaining exams (in the second part) to the earlier timeslots (in the second part) based on the feasibility.

Step 3: Find the missing exams in the Offspring 1 and reschedule them in the second part based on the feasibility. Hence, the feasible Offspring 1 is generated.

Step 4: Repeat **Step 1-3** for Offspring 2.

Figure 4 shows a possible example of the feasible solutions for Offspring 1 and 2.



Offspring 1:
(Feasible)

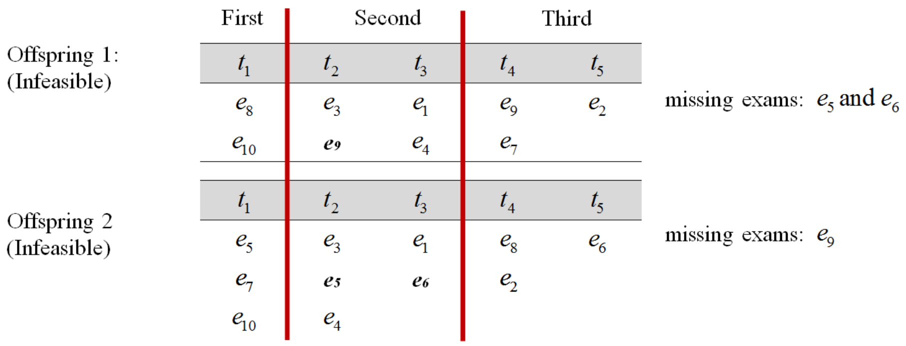
t_1	t_2	t_3	t_4	t_5
e_8	e_3	e_1	e_2	e_9
e_{10}	e_5	e_6	e_7	
	e_4			

Offspring 2:
(Feasible)

t_1	t_2	t_3	t_4	t_5
e_5	e_3	e_1	e_2	e_8
e_7	e_9	e_4	e_6	
e_{10}				

Figure 4: Feasible Offspring 1 and Offspring 2 of M1PC

In the modified two-point crossover (M2PC), two crossover points are randomly selected within the two parents to form three separate parts of timetable. We refer back to Figure 2 for the illustration purposes of this crossover. For instance, the first crossover point is between gene 1 and gene 2 while the second crossover point is between gene 3 and gene 4. Then, the first and the third part of Parent 1 and 2 are copied into Offspring 1 and 2 while the second part of Offspring 1 and 2 are then filled with the second part of Parent 2 and 1 respectively. Figure 5 below shows the offspring that are produced after the interchanging process.



Offspring 1: (Infeasible)

First	Second		Third	
t_1	t_2	t_3	t_4	t_5
e_8	e_3	e_1	e_9	e_2
e_{10}	e_9	e_4	e_7	

missing exams: e_5 and e_6

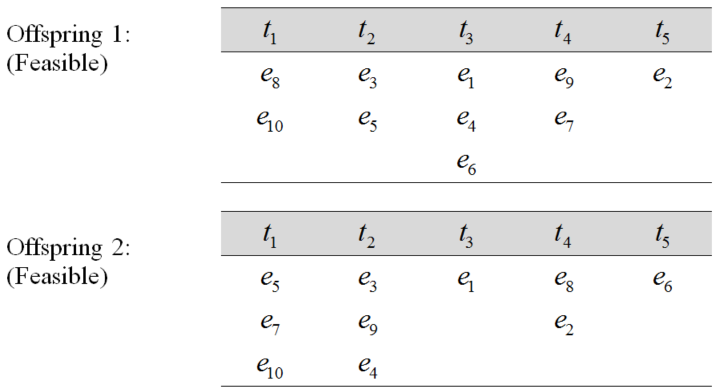
Offspring 2 (Infeasible)

First	Second		Third	
t_1	t_2	t_3	t_4	t_5
e_5	e_3	e_1	e_8	e_6
e_7	e_5	e_6	e_2	
e_{10}	e_4			

missing exams: e_9

Figure 5: Infeasible Offspring 1 and Offspring 2 of M2PC

There are cases after the interchanging, both the offspring do not represent feasible solutions. For Offspring 1, duplication of exam has occurred in exam e_9 while exams e_5 and e_6 are missing. Similarly in Offspring 2, the duplicated exams are e_5 and e_6 while the missing exam is e_9. Therefore, the similar steps as in M1PC are taken in order to produce feasible solutions. Figure 6 shows a possible example of feasible solutions for Offspring 1 and Offspring 2.



Offspring 1: (Feasible)

t_1	t_2	t_3	t_4	t_5
e_8	e_3	e_1	e_9	e_2
e_{10}	e_5	e_4	e_7	
		e_6		

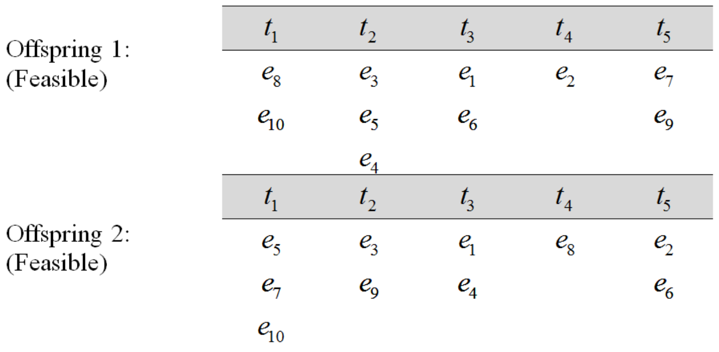
Offspring 2: (Feasible)

t_1	t_2	t_3	t_4	t_5
e_5	e_3	e_1	e_8	e_6
e_7	e_9		e_2	
e_{10}	e_4			

Figure 6: Feasible Offspring 1 and Offspring 2 of M2PC

The modified position-based crossover (MPBC) has the similar concept as M1PC. The MPBC aims at preserving the schedules of the exams as much as possible. It is best illustrated by using an example based on Figure 2. Assumed the crossover point between Parent 1 and 2 is between gene 3 and gene 4. The first part of Parent 1 and 2 are copied to Offspring 1 and 2 while the second part of the Offspring 1 and 2 are copied from Parent 2 and 1 respectively. Both offspring are identical to M1PC as shown in Figure 3.

The difference between the M1PC and MPBC is on the step of rescheduling the exams in the second part of Offspring 1 and 2 in order to produce feasible solutions while preserving the schedules of the exams. Instead of reschedule the remaining exams in the second part to the earlier timeslots as in **Step 2** of M1PC, the MPBC will keep the remaining exams in the second part of the offspring at the same position in the timeslots. Figure 7 shows a possible example of feasible solutions for Offspring 1 and 2.



Offspring 1:
(Feasible)

t_1	t_2	t_3	t_4	t_5
e_8	e_3	e_1	e_2	e_7
e_{10}	e_5	e_6		e_9
	e_4			

Offspring 2:
(Feasible)

t_1	t_2	t_3	t_4	t_5
e_5	e_3	e_1	e_8	e_2
e_7	e_9	e_4		e_6
e_{10}				

Figure 7: Feasible Offspring 1 and Offspring 2 of MPBC

A computational comparison is also made among four types of standard mutation operators. The best operator that gives the best result will be adopted to be the mutation operator in the proposed PGA. These mutation operators are inverse, insertion, exchange and scramble mutations. Lastly, in replacement strategies, three types of standard replacement strategies are tested to identify the best replacement strategy that gives the best result to be the replacement strategy in the proposed PGA. These replacement strategies are: en-bloc, elitism and steady-state replacement.

A series of initial computational investigation is performed based on three datasets extracted from the Carter's benchmark problems. These datasets are chosen based on the total number of registered students (i.e. low, medium and high) at their respective institutions. The datasets are sta-83 (611 students), lse-91 (2726 students) and kfu-93 (5349 students). For each computational experiment of the genetic operators, each dataset is run for five times to get the average values. Note that the population size for the proposed algorithm is kept constant at 500 chromosomes and the execution of the PGA is halted once the algorithm reached 100 generations. These are the standard values used for the GAs in the literature. At each stage of the development, the 'winner' of the experiment will be directly adopted into the PGA before proceed to the next experiment. The tables of computational results for the genetic operators of PGA are given in Appendix A. For each table of results, the first two columns give the dataset and the number of run. The next few sets of three columns refer to the genetic operators being tested in each experiment. For each genetic operator, the entries in the first column report the penalty cost (Equation 1), the second column report the total timeslot, and the last column give the CPU time (in second) of 100 generations. For each dataset, the final row gives the average value of five runs. The bold face figures represent the best solution obtained for each dataset. Based on the initial investigation, the final scheme of the best performed genetic operators for the proposed PGA is presented in Figure 8.

```
BEGIN
        INITIALIZE population using low-level heuristic construction
        EVALUATE each individual
Repeat
        SELECT parents using tournament selection
        RECOMBINE pairs of parents to produce new offspring using M2PC
        MUTATE the resulting offspring using scramble mutation
        EVALUATE each offspring
        REPLACEMENT for new population using steady-state replacement
Until
        TERMINATION-CONDITION is satisfied
END
```

Figure 8: Final Scheme of the Best Performed Genetic Operators for the Proposed PGA

4. Hybrid Genetic Algorithm

This section describes the improvement made on the PGA by proposing the improvement phase of the algorithm. The PGA will hybridise with three new local optimisation techniques, which will make up the HGA, to improve the solution found by PGA. The techniques will make alterations to the original timetables. This involves moving or swapping a scheduled exams, e_k, or interchanging the timeslot, t_i. First technique focuses on inserting a scheduled exam to a new timeslot, second technique concerns with swapping of two scheduled exams between two different timeslots, and the third technique concerns with interchanging the timeslots in the timetable. We introduce new explicit equations for each technique. During the implementation of the HGA, one of the three new local optimisation techniques will be randomly selected to be executed after the mutation operator is performed. This is due to the expensive computational runtime of the local optimisation technique for large dataset. With the introduction of the randomisation in selecting the new local optimisation technique in the HGA, the algorithm will performed better than a PGA with a fixed local optimisation technique (see, Table 4, Section 5.3). The descriptions of each technique are given in the following subsections.

4.1 Move Exam (ME)

In this approach, the algorithm will move an already scheduled exam, e_k, (in timeslot t_q) to a new position (in timeslot t_p) if it results in reducing the soft constraints cost and at the same time improve the overall quality of the timetable. To achieve this, a new Exam/Timeslot Conflict Matrix is constructed where each element, $\sigma_{k,i}$, stores the value of the total number of common students between the exam e_k with all other exams scheduled in a timeslot t_i. Note that, a timetable is feasible if $\sigma_{k,i} = 0, (k = 1, 2, \cdots, E)$, where E is the total number of exams to be scheduled. In other words, the value of $\sigma_{k,i}$ is obtained from the following equation:

$$\sigma_{k,i} = \sum_{s=1}^{n_i} c_{k,e_{i,s}}, \tag{3}$$

where

$e_{i,s}$ = exam e_s $(s = 1, 2, \cdots, n_i)$ scheduled in timeslot t_i, and n_i is the total number of exams in the timeslot t_i.

$c_{k,e_{i,s}}$ = number of common students between exam e_k with the exam $e_{i,s}$ scheduled in timeslot t_i.

New equations are introduced to calculate the cost of inserting the exam e_k to the new timeslots, t_p in the timetable. Let A be the total penalty cost of the timetable without exam e_k been scheduled in the timetable. Equation 4 below calculate the total penalty cost, Q of the original timetable with exam e_k is scheduled in timeslot t_q.

$$Q = A + \sum_{m=1}^{5} 2^{5-m} \left(\sigma_{k,q+m} + \sigma_{k,q-m}\right), \tag{4}$$

where

$\sigma_{k,q+m}$ and $\sigma_{k,q-m}$ represent the total number of common students between exam e_k and all other exams scheduled in timeslot t_{q+m} and t_{q-m} respectively.
$q + m \leq T$ and $q - m \geq 1$.

The exam e_k is inserted into a new timeslot t_p, if and only if, $\sigma_{k,p} = 0$. The new total penalty cost, Q_1 is given by the following equation:

$$Q_1 = A + \sum_{m=1}^{5} 2^{5-m} \left(\sigma_{k,p+m} + \sigma_{k,p-m}\right), \tag{5}$$

where

$\sigma_{k,p+m}$ and $\sigma_{k,p-m}$ represent the total number of common students between exam e_k and all other exams scheduled in timeslot t_{p+m} and t_{p-m} respectively.
$p + m \leq T$ and $p - m \geq 1$.

By subtracting Equation 4 from Equation 5, we have:

$$Q_1 - Q = \sum_{m=1}^{5} 2^{5-m} \left(\sigma_{k,p+m} + \sigma_{k,p-m} - \sigma_{k,q+m} - \sigma_{k,q-m}\right), \tag{6}$$

where

$p + m \leq T$, $p - m \geq 1$, $q + m \leq T$ and $q - m \geq 1$.

If $Q_1 - Q < 0$ (the total penalty cost when exam e_k scheduled in the timeslot t_p is lower than the total penalty cost in the timeslot t_q), then the exam e_k is inserted into the timeslot t_p and this will lead to a decrease in the total penalty cost of the entire timetable. The algorithm is executed based on the first improvement move, i.e. the algorithm is halted once an improved solution is found. If there is no improving move found after all the exams are checked, the ME algorithm will be terminated and it is assumed that the timetable is optimum based on this technique.

4.2 Swap Exam (SE)

During the implementation of this technique, two exams from two different timeslots will swap their position in the timetable if it leads to a decrease in the soft constraint cost. Let exams e_i and e_j are initially scheduled in the timeslot t_p and t_q respectively. A^* be the total penalty cost of the timetable without exams e_i and e_j been scheduled in the timetable. Equation 7 calculates the total penalty cost, R of the original timetable with exam e_i and e_j are scheduled in the timeslots t_p and t_q respectively:

$$R = A^* + \sum_{m=1}^{5} 2^{5-m} \left(\sigma_{i,p+m} + \sigma_{i,p-m} + \sigma_{j,q+m} + \sigma_{j,q-m}\right), \tag{7}$$

where

$\sigma_{i,p+m}$ and $\sigma_{i,p-m}$ represent the total number of common students between exam e_i and all other exams scheduled in timeslots t_{p+m} and t_{p-m} respectively.
$\sigma_{i,q+m}$ and $\sigma_{i,q-m}$ represent the total number of common students between exam e_i and all other exams scheduled in timeslots t_{q+m} and t_{q-m} respectively.
$p+m \leq T, p-m \geq 1, q+m \leq T$, and $q-m \geq 1$.

Now, assumed that these two exams can be swapped, i.e. $\sigma_{i,q} = \sigma_{j,p} = 0$. Then, the new total penalty cost, R_1 after the swap where exams e_i and e_j are scheduled in the timeslots t_q and t_p respectively is given in the following equation:

$$R_1 = A^* + \sum_{m=1}^{5} 2^{5-m} \left(\sigma_{i,q+m} + \sigma_{i,q-m} + \sigma_{j,p+m} + \sigma_{j,p-m}\right). \tag{8}$$

Similar to the ME algorithm, by subtracting Equation 7 from the Equation 8, the result is shown below:

$$R_1 - R = \sum_{m=1}^{5} 2^{5-m} \left(\sigma_{i,q+m} + \sigma_{i,q-m} + \sigma_{j,p+m} + \sigma_{j,p-m} - \sigma_{i,p+m} - \sigma_{i,p-m} - \sigma_{j,q+m} - \sigma_{j,q-m}\right), \tag{9}$$

where

$p+m \leq T, p-m \geq 1, q+m \leq T$, and $q-m \geq 1$.

If $R_1 - R < 0$, then the two exams (e_i and e_j) between timeslots t_p and t_q are swapped. The algorithm is executed based on the first improving move. Similar to ME technique, the timetable is assumed to be optimal based on the SE technique if there is no improving found after all the exams are tested.

4.3 Interchange Timeslot (IT)

This approach involves interchanging of two timeslots in the timetable if it results in an improvement in the quality of the timetable (i.e. reduces the soft constraints cost). Note that this approach is differs from other mutation operators where the interchanging of timeslots in mutation occurred at random. For this purpose, a new Timeslots Conflict Matrix is introduced, where each element $\mu_{i,j}$ stores the value of the total number of common students between two timeslots, t_i and t_j. Note that, $\mu_{i,j} = 0$ if $i = j$ or both timeslots do not have students in common. This approach differs from the previous two techniques where the feasibility check on the exams is not required since all exams within a timeslot are interchanged with all exams from another timeslot. We introduced the following equations to calculate the cost of interchanging these two timeslots. Equation 10 below calculates the total penalty cost, P of the complete timetable before interchanging t_i and t_j.

$$\begin{aligned}
&P = 16\left(\mu_{1,2}+\mu_{2,3}+\cdots+\mu_{i-1,i}+\mu_{i,i+1}+\cdots+\mu_{j-1,j}+\mu_{j,j+1}+\cdots+\mu_{T-1,T}\right)+\\
&8\left(\mu_{1,3}+\mu_{2,4}+\cdots+\mu_{i-2,i}+\mu_{i-1,i+1}+\mu_{i,i+2}+\cdots+\mu_{j-2,j}+\mu_{j-1,j+1}+\mu_{j,j+2}\cdots+\mu_{T-2,T}\right)+\\
&4\left(\mu_{1,4}+\mu_{2,5}+\cdots+\mu_{i-3,i}+\cdots+\mu_{i,i+3}+\cdots+\mu_{j-3,j}+\cdots+\mu_{j,j+3}+\cdots+\mu_{T-3,T}\right)+\\
&2\left(\mu_{1,5}+\mu_{2,6}+\cdots+\mu_{i-4,i}+\cdots+\mu_{i,i+4}+\cdots+\mu_{j-4,j}+\cdots+\mu_{j,j+4}+\cdots+\mu_{T-4,T}\right)+\\
&\left(\mu_{1,6}+\mu_{2,7}+\cdots+\mu_{i-5,i}+\cdots+\mu_{i,i+5}+\cdots+\mu_{j-5,j}+\cdots+\mu_{j,j+5}+\cdots+\mu_{T-5,T}\right).
\end{aligned} \tag{10}$$

Now, assumed that the timeslots t_i and t_j is interchanged. The new penalty cost function, P_1 after the interchange is given as:

$$\begin{aligned}
&P_1 = 16\left(\mu_{1,2}+\mu_{2,3}+\cdots+\mu_{i-1,i}+\mu_{j,i+1}+\cdots+\mu_{j-1,i}+\mu_{i,j+1}+\cdots+\mu_{T-1,T}\right)+\\
&8\left(\mu_{1,3}+\mu_{2,4}+\cdots+\mu_{i-2,j}+\mu_{i-1,i+1}+\mu_{j,i+2}+\cdots+\mu_{j-2,i}+\mu_{j-1,j+1}+\mu_{i,j+2}\cdots+\mu_{T-2,T}\right)+\\
&4\left(\mu_{1,4}+\mu_{2,5}+\cdots+\mu_{i-3,j}+\cdots+\mu_{j,i+3}+\cdots+\mu_{j-3,i}+\cdots+\mu_{i,j+3}+\cdots+\mu_{T-3,T}\right)+\\
&2\left(\mu_{1,5}+\mu_{2,6}+\cdots+\mu_{i-4,j}+\cdots+\mu_{j,i+4}+\cdots+\mu_{j-4,i}+\cdots+\mu_{i,j+4}+\cdots+\mu_{T-4,T}\right)+\\
&\left(\mu_{1,6}+\mu_{2,7}+\cdots+\mu_{i-5,j}+\cdots+\mu_{j,i+5}+\cdots+\mu_{j-5,i}+\cdots+\mu_{i,j+5}+\cdots+\mu_{T-5,T}\right).
\end{aligned} \tag{11}$$

By subtracting Equation 10 from Equation 11, we have:

$$P_1 - P = \sum_{m=1}^{5} 2^{5-m}\left(\mu_{i-m,j}+\mu_{j,i+m}+\mu_{j-m,i}+\mu_{i,j+m}-\mu_{i-m,i}-\mu_{i,i+m}-\mu_{j-m,j}-\mu_{j,j+m}\right), \tag{12}$$

where

$i > j, i-m \geq 1, i+m \leq T, j-m \geq 1$, and $j+m \leq T$.

If $P_1 - P < 0$, we can conclude that the total penalty cost is improved by interchanged the timeslots t_i and t_j. The steps are repeated until no more interchange that can lower the penalty cost function. If there is no improving move after all the timeslots are checked, the IT algorithm is terminated and we assumed that the timetable is optimum based to this technique.

5. Computational Experiments

5.1 Experimental Designs

All proposed algorithms are coded in ANSI-C using Microsoft Visual C++ 6.0 as the compiler and run on a laptop computer running on Window 7 as the operating system with Intel Core i5, 4 GB memory. To measure and compare the solution quality of the proposed algorithms, the proximity (penalty) cost function given in Equation 1 is used. Note that the population size for all proposed algorithms is kept constant at 500 chromosomes, a standard value used for the GAs in the literature.

The problem instances are taken from the Carter benchmark dataset as shown in Table 2. This dataset is introduced by Carter et al. (1996) which consist of 12 real-world exam timetabling problems from three Canadian high schools, five Canadian, one United Kingdom and one mid-east universities. Various important information to construct the timetable such as the total number of periods available for each dataset, the total number of exams involved, total number of students registered for the exams and the conflict density can be obtained from this table.

Table 2: The Carter Benchmarks Dataset

Dataset	Institution	Timeslots	Exams	Students	Enrolments	Density
sta-83	St. Andrew's Junior High School, Toronto, Canada	13	139	611	5751	0.14
yor-83	York Mills Collegiate Institute, North York, Canada	21	181	941	6034	0.29
ear-83	Earl Haig Collegiate Secondary Scholle, North York, Canada	24	190	1125	8109	0.27
hec-92	Ecole des Hautes Etudes Commerciales de Montrea, Quebec, Canada	18	81	2823	10632	0.42
lse-91	London School of Economics, London, UK	18	381	2726	10918	0.06
ute-92	Faculty of Engineering, University of Toronto, Canada	10	184	2750	11793	0.08
tre-92	Trent University, Peterborough, Canada	23	261	4360	14901	0.18
kfu-93	King Fahd University of Petroleum and Minerals, Dharan, Saudi Arabia	20	461	5349	25113	0.06
rye-93	Ryerson University, Toronto, Canada	23	486	11483	45051	0.08
car-91	Carleton University, Ottawa, Canada	35	682	16925	56877	0.13
car-92	Carleton University, Ottawa, Canada	32	543	18419	55522	0.14
uta-92	Faculty of Arts and Sciences, University of Toronto, Toronto, Canada	35	622	21266	58976	0.13

5.2 Computational Results of PGA

In this subsection, we first present the computational results of the proposed PGA for UETP. For statistical significance, 30 runs were performed for each dataset. The execution of the PGA will be terminated when the penalty cost failed to improve in 30 consecutive generations. Table 3 shows the best, average and the worst penalty costs for 12 datasets produced from the proposed PGA. Note that all computational experiments of PGA in this subsection produced the same number of timeslots for each run of each dataset as reported in Table 2. Hence, the results for the number of timeslots for each dataset are omitted. The first column gives the name of the dataset. The next three columns refer to the best, average and worst penalty cost values for each dataset respectively. The entries in the average column reports the average penalty cost of 30 runs.

Table 3: Computational Results of PGA for Carter Benchmark Dataset

Dataset	**Best**	**Average**	**Worst**
sta-83	162.85	163.45	164.60
yor-83	39.91	45.44	47.27
ear-83	30.99	43.58	47.71
lse-91	13.73	14.91	15.76
ute-92	30.57	32.14	34.09
hec-92	12.33	12.96	13.67
tre-92	8.11	10.49	11.26
kfu-93	16.80	17.47	18.30
rye-93	12.99	13.90	14.79
car-91	6.63	6.95	7.12
car-92	5.70	5.89	6.12
uta-92	4.40	4.61	4.77

From the results shown, the proposed PGA managed to get the solutions with a small gap between the best and the worst solutions over 30 runs except for dataset ear-83 and tre-92. This shows that the proposed PGA is robust in finding the solutions. However, some modifications are needed in order to improve the final solutions so that our proposed PGA is comparable with other algorithms proposed in the literature. To achieve this, the PGA is hybridised with the three local optimisation techniques proposed in Section 4, which make up the HGA.

5.3 Computational Results of HGA

In this subsection, we report the computational experiments of the proposed HGA. The computational results are presented in Table 4. The performance of the three new local optimisation techniques, i.e. ME, SE, and IT are tested by combining with the PGA one by one. Their performances are also compared with the HGA, where in every generation of the HGA algorithm, one of the three new techniques will be randomly selected to be executed after the mutation operator is performed. The first column gives the dataset. The next five sets of two columns refer to the PGA, PGA with ME, PGA with SE, PGA with IT, and HGA algorithms respectively. For each algorithm, the entries in the first column report the best penalty cost and the second column report the average penalty cost. The bold face figures represent the best and average penalty cost values obtained for each dataset by the algorithms.

From the computational results, the PGAs with fixed local optimisation technique achieved a mixed degree of success as compared to the PGA. In general, the PGAs with fixed optimisation technique improved the solutions slightly in most dataset. When compared the combination of PGA with a fixed local optimisation techniques, the combination of PGA with IT gave better penalty cost values. The improvements of the solutions quality are more significant for the HGA as compared to the PGA where an element of the randomisation is introduced in selecting a local optimisation technique to be executed after the mutation operator is performed. The results proved the robustness of the HGA when dealing with the Carter benchmark dataset. Hence, we can conclude that HGA is the best algorithm where it produced the best penalty cost for all datasets. HGA also showed the best average's penalty cost value for all the datasets except for yor-83.

Table 4: Computational Results of PGA, PGA with fixed Local Optimisation Techniques and HGA

Dataset	**PGA**		**PGA + ME**		**PGA + SE**		**PGA + IT**		**HGA**	
	Best	Average	Best	Average	Best	Average	Best	Average	Best	Average
sta-83	162.85	163.45	160.03	163.02	163.22	163.88	159.09	159.23	**158.94**	**159.05**
yor-83	39.91	45.44	39.28	42.12	42.17	44.05	38.41	**39.32**	**38.28**	39.39
ear-83	41.99	43.58	43.07	41.52	44.57	45.75	40.06	41.07	**36.97**	**37.51**
lse-91	13.73	14.91	12.77	13.42	13.19	14.10	12.77	13.14	**11.67**	**11.95**
ute-92	30.57	32.14	28.19	29.87	30.27	32.14	28.11	29.06	**26.71**	**27.44**
hec-92	12.33	12.96	12.06	12.76	12.25	13.05	11.02	11.29	**10.79**	**11.01**
tre-92	9.51	10.49	9.01	9.60	9.37	10.24	9.34	9.57	**8.64**	**8.94**
kfu-93	16.80	17.47	15.67	16.05	16.07	17.13	15.40	15.81	**14.85**	**15.25**
rye-93	12.99	13.90	11.73	12.45	11.52	13.10	11.43	11.88	**9.83**	**10.14**
car-91	6.63	6.95	5.69	5.97	6.41	6.87	6.16	6.23	**5.43**	**5.53**
car-92	5.70	5.89	4.88	5.09	5.56	5.82	5.08	5.19	**4.54**	**4.66**
uta-92	4.40	4.61	3.86	4.00	4.42	4.57	4.12	4.16	**3.36**	**3.68**

5.4 Comparisons of Different Methodologies

To our knowledge, in the literature, only one study applied GA to the Carter benchmark and it was conducted by Pillay and Banzhaf (2010). Another study by Pillay (2012) applied the EA to the Carter dataset. Thus, in this final subsection, to expand the comparison, we first compare the performance of proposed HGA with other GA/EA in the literature that applied to the same Carter datasets. Then, for the second comparison, the performance of HGA is compared with the benchmark results from the literature using different methodologies. The computational results of the first comparison are presented in Table 5. The first column gives the dataset. For each algorithm, the entries report the best penalty cost value. The bold face figures represent the best penalty cost value obtained for each dataset by the algorithms. The comparison is made based on the penalty cost value of the best solution (timetable) found by the algorithms. In this study, the comparison of the CPU runtime is not considered because for different studies, varying computing powers are used to run the algorithms.

Table 5: Comparative Computational Results of HGA and other EA Approaches

Dataset	**HGA**	**Pillay & Banzhaf (2010)**	**Pillay (2012)**		
		IGA	CEA	VHC	NHC
sta-83	158.94	157.81	157.69	157.82	**157.39**
yor-83	**38.28**	39.33	39.63	40.00	39.84
ear-83	36.97	**35.87**	35.95	35.94	36.64
lse-91	11.67	10.89	**10.76**	10.83	10.81
ute-92	**26.71**	27.24	26.95	27.13	27.16
hec-92	**10.79**	11.50	11.27	11.21	11.26
tre-92	8.64	8.38	8.43	**8.37**	8.48
kfu-93	14.85	14.37	**14.12**	14.13	14.21
rye-93	9.83	9.30	**9.23**	**9.23**	9.25
car-91	5.43	**4.92**	4.95	4.95	4.93
car-92	4.54	4.22	4.22	4.19	**4.18**
uta-92	3.36	3.35	3.33	3.37	**3.32**

From the results presented, it can be seen that the HGA performed well for yor-83, ute-92 and hec-92 datasets. For other datasets, the results obtained by HGA are comparable to the performance of other EA approaches. There is no clear winner on which GA/EA approach that performed the best. Each approaches obtained the best solution for at most three dataset.

For the second comparison, Table 6 compares the best results obtained from the proposed HGA in comparison with other published results on benchmark datasets from the literature. The first column gives the dataset. For each algorithm, the entries report the best penalty cost value. The results that are underperformed as compared to the HGA are highlighted in red.

From the results shown in Table 6, although the proposed HGA does not yield the best result for any of the benchmark dataset, but its performance is clearly competitive to the others methodologies. It is also worth pointing out that the proposed HGA managed to produce feasible solutions for all 12 datasets. In addition to that, the HGA produces better penalty cost values when compared to the TS implementation by Di Gaspero and Schaerf (2001) for all 12 datasets. Our proposed HGA also has outperformed 11 out of the 12 datasets for three other methodologies which are construction approach based on adaptive decomposition and ordering by Abdul-Rahman et al. (2014), fuzzy multiple heuristic orderings by Asmuni et al. (2005) and genetic programming for auction based scheduling by Bader-El-Den and Fatima (2010). However, Bader-El-Den and Fatima (2010) did not solve the ute-92 and rye-93 datasets. The HGA also produces better penalty cost value than Burke et al. (2009) on seven datasets and has the same results for the other three datasets. The methods of GHH by Burke et al. (2007) outperformed our method on two datasets. For both methodologies implemented by Burke ((Burke et al., 2007) and (Burke et al., 2009)), they did not solve the rye-93 dataset. For other methodologies, in general, the proposed HGA produces better results on at least two of the datasets.

Table 6: Comparative Computational Results of HGA and other Methodologies

Dataset	**HGA**	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
sta-83	**158.9**	**159.0**	**160.8**	**162.3**	**160.4**	**159.2**	158.9	134.9	158.2	157.3	158.2	157.0	158.2
yor-83	**38.3**	36.2	**41.0**	**45.2**	**40.7**	36.2	**42.2**	37.5	**41.4**	**45.6**	**41.6**	36.2	36.2
ear-83	**37.0**	36.0	**45.7**	**40.6**	37.0	34.9	36.5	34.8	**37.9**	**41.2**	36.7	34.7	29.3
lse-91	**11.7**	**11.9**	**15.5**	**12.0**	**12.1**	10.2	**12.1**	**14.7**	11.3	**13.0**	10.9	10.0	9.6
ute-92	**26.7**	26.0	**29.0**	**27.9**	**27.8**	26.0	26.6	25.4	**28.0**	-	26.7	25.9	24.4
hec-92	**10.8**	10.8	**12.4**	**12.4**	**11.8**	10.3	**11.8**	10.8	**12.3**	**13.0**	**11.6**	10.7	9.2
tre-92	**8.6**	8.5	**10.0**	**9.7**	**8.7**	8.4	**9.0**	**8.7**	**8.8**	**9.3**	8.5	7.9	**9.4**
kfu-93	**14.9**	**15.2**	**18.0**	**16.2**	**15.8**	13.5	**15.5**	14.1	**15.2**	**15.9**	14.2	13.0	13.8
rye-93	**9.8**	-	-	**10.3**	**10.4**	8.7	-	-	-	-	9.4	**11.0**	6.8
car-91	**5.4**	4.8	**6.2**	5.3	5.3	5.2	5.4	5.4	5.4	**7.0**	-	4.8	**6.6**
car-92	**4.5**	4.1	**5.2**	**4.9**	**4.6**	4.4	4.5	4.4	4.5	**5.8**	-	3.9	**6.0**
uta-92	**3.4**	**3.6**	**4.2**	**3.5**	**3.6**	**3.6**	**3.5**	-	**3.9**	**3.8**	-	3.1	**3.5**

Note: '-' = no solution reported
(1) Multi Start Two Phase Approach, by Abdullah and Burke (2006)
(2) Tabu Search, by Di Gaspero and Schaerf (2001)
(3) Construction Approach based on Adaptive Decomposition and Ordering, by Abdul-Rahman et al. (2014)
(4) Fuzzy Multiple Heuristic Orderings, by Asmuni et al. (2005)
(5) Ahuja-Orlin's large neighbourhood search approach, by Abdullah et al. (2007)
(6) Adaptive Selection of Heuristics within a GRASP, by Burke et al. (2009)
(7) GRASP, by Casey and Thompson (2003)
(8) Graph-based Hyper-heuristic, by Burke et al. (2007)
(9) Genetic Programming, by Bader-El-Den and Fatima (2010)
(10) Genetic Programming Approach to the Generation of Hyper-Heuristics, by Pillay and Banzhaf (2007)
(11) Honey-Bee Mating Optimization (ETP-HBMO), by Sabar et al. (2009)
(12) The Sequential Heuristic Construction Method, by Caramia et al. (2001)

6. Conclusions

The overall goal of this paper is to investigate the strength of HGA in solving UETP. This is done in two phases: construction phase (generate feasible timetables) and improvement phase (improve the quality of timetable found in the first phase by reducing the penalty cost value of the problem). After the initial population is generated, the best combination of the proposed PGA operators are examined first before the PGA is hybridised with three new local optimisation techniques to construct the HGA.

In the construction phase, the most challenging part is to construct a feasible solution (timetable) with the required number of timeslots where feasible timetable is important to avoid students from sitting for more than one exam simultaneously. In this paper, two types of construction methods which are random initialisation and low-level heuristic construction method were used to generate the feasible timetables in the initial population. The computational results showed that the low-level heuristic construction method performed better than random initialisation method.

In the improvement phase, a HGA with three new local optimisation techniques is used to spread the students' exams throughout the available timeslots given by the institution in order to give extra gaps between the exams. In the first technique, an exam is moved to another timeslot, if and only if, the movement caused a reduction in the penalty cost function. Second technique considered swapping two exams whereas the third technique is interchanging the timeslots until no further improvement to the timetable can be achieved. New explicit equations are introduced in this paper to calculate the effect of altering the exams/ timeslots in the timetables.

The proposed HGA is demonstrated on the 12 uncapacitated UETP of the Carter's benchmark datasets to measure the efficiency of the algorithm. The computational experiment found that by applying one local optimisation technique randomly from these three new local optimisation techniques at each generation gave better penalty cost value which is calculated in terms of how well the exams with common students are spaced. We compared the proposed HGA with other metaheuristics which used the same benchmark datasets from the literature. The results shown that the proposed HGA outperformed some of the metaheuristic approaches and comparable within the range to most of the metaheuristic approaches.

References

Abdul-Rahman, S., Burke, E. K., Bargiela, A., McCollum, B., and Özcan, E. (2014). A Constructive Approach to Examination Timetabling based on Adaptive Decomposition and Ordering. *Annals of Operations Research*, 218(1):3–21.

Abdullah, S., Ahmadi, S., Burke, E. K., and Dror, M. (2007). Investigating Ahuja-Orlin's large Neighbourhood Search Approach for Examination Timetabling. *OR Spectrum*, 29(2):351–372.

Abdullah, S. and Burke, E. K. (2006). A Multi-Start Very Large Neighbourhood Search Approach with Local Search Methods for Examination Timetabling. In *Intervention Centred on Adolescents' Physical Activity and Sedentary Behaviour (ICAPS)*, pages 334–347.

Abounacer, R., Boukachour, J., Dkhissi, B., and El Hilali Alaoui, A. (2010). A Hybrid Ant Colony Algorithm for The Exam Timetabling Problem. *Revue African Research Journal In Computer Science (ARIMA)*, 12:15–42.

Asmuni, H. (2008). *Fuzzy Methodologies for Automated University Timetabling Solution Construction and Evaluation.* PhD thesis, University of Nottingham.

Asmuni, H., Burke, E., and J.M., G. (2005). Fuzzy Multiple Ordering Criteria for Examination Timetabling. In *Proceedings of 5th International Conference on the Theory and Practice of Automated Timetabling (PATAT 2004)*, pages 147–160.

Bader-El-Den, M. and Fatima, S. (2010). Genetic Programming for Auction Based Scheduling. In *Genetic Programming*, pages 256–267. Springer: Berlin.

Burke, E., Meisels, A., Petrovic, S., and Qu, R. (2007). A Graph-based Hyper-Heuristic for Timetabling Problems. *European Journal of Operational Research*, 176(1):177–192.

Burke, E. K., Elliman, D. G., Ford, P. H., and Weare, R. F. (1996a). Examination Timetabling in British Universities: a survey. In *Practice and Theory of Automated Timetabling*, pages 76–90. Springer: Berlin.

Burke, E. K. and Newall, J. P. (2004). Solving Examination Timetabling Problems Through Adaptation of Heuristic Orderings. *Annals of Operational Research*, 129:107–134.

Burke, E. K., Newall, J. P., and Weare, R. F. (1996b). A Memetic Algorithm for University Exam Timetabling. In *Practice and Theory of Automated Timetabling*, pages 241–250. Springer: Berlin.

Burke, E. K., Qu, R., and Soghier, A. (2009). Adaptive Selection of Heuristics within a GRASP for Exam Timetabling Problems. In *Proceedings of the 4th Multidisciplinary International Scheduling: Theory and Applications 2009 (MISTA 2009)*, pages 93–104, Ireland, Dublin.

Caramia, M., Dell'Olmo, P., and Italiano, G. F. (2001). New Algorithms for Examination Timetabling. In *Proceedings of the 4th International Workshop WAE 2000*, pages 230–241. Springer: Berlin.

Carter, M. W. (1986). A Survey of Practical Applications of Examination Timetabling Algorithms. *Operations Research*, 34:193–202.

Carter, M. W., Laporte, G., and Lee, S. Y. (1996). Examination Timetabling: Algorithmic Strategies and Applications. *Journal of Operational Research Society*, 47(3):373–383.

Casey, S. and Thompson, J. (2003). GRASPing The Examination Scheduling Problem. In *Practice and Theory of Automated Timetabling*, volume IV, pages 232–244. Springer: Berlin.

Chu, S. C. and Fang, H. L. (1999). Genetic Algorithms vs. Tabu Search in Timetable Scheduling. In *Proceedings of the 3rd International Conference on Knowledge-Based Intelligent Information Engineering Systems*, pages 492–495, Adelaide, Australia. IEEE Press.

Corne, D., Ross, P., and Fang, H. (1994). Evolutionary Timetabling: Practice, Prospects and Work in Progress. In *Proceedings of UK Planning and Scheduling SIG Workshop*.

Di Gaspero, L. and Schaerf, A. (2001). Tabu Search Techniques for Examination Timetabling. In *Practice and Theory of Automated Timetabling*, volume III, pages 104–117. Springer: Berlin.

Kendall, G. and Hussin, N. M. (2005). An Investigation of a Tabu Search Based Hyper-Heuristic for Examination Timetabling. In Kendall, G., Burke, E., and Petrovic, S., editors, *Selected papers from Multidisciplinary International Scheduling: Theory and Applications*, pages 309–328.

Pillay, N. (2012). Evolving Hyper-Heuristics for The Uncapacitated Examination Timetabling Problem. *Journal of the Operational Research Society*, 63:47–58.

Pillay, N. and Banzhaf, W. (2007). A Genetic Programming Approach to the Generation of Hyper-Heuristics for the Uncapacitated Examination Timetabling Problem. In *Progress in Artificial Intelligence*, pages 223–234. Springer: Berlin.

Pillay, N. and Banzhaf, W. (2008). A Developmental Approach to the Examination Timetabling Problem. In *Parallel Problem Solving from Nature*, volume X, pages 276–285. Springer: Berlin.

Pillay, N. and Banzhaf, W. (2010). An Informed Genetic Algorithm for The Examination Timetabling Problem. *Applied Soft Computing*, 10:457–467.

Reis, L. P. and Oliveira, E. (1999). Constraint Logic Programming using Set Variables for solving Timetabling Problems. In *Proceedings 12th International Conference on Applications of Prolog.*

Sabar, N. R., Ayob, M., and Kendall, G. (2009). Solving Examination Timetabling Problems using Honey-Bee Mating Optimization (ETP-HBMO). In *Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2009)*, pages 399–408, Ireland, Dublin.

Appendix A

Table 7: Comparative Computational Results of Random Initialisation and Low-Level Heuristic Construction Method

Dataset	**Run**	**Random**			**Low-Level Heuristic**		
		Penalty Cost	Total Timeslot	CPU Time (sec)	Penalty Cost	Total Timeslot	CPU Time (sec)
sta-83	I	192.06	14	0.06	186.92	13	0.62
	II	191.89	14	0.10	187.25	13	0.55
	III	192.34	14	0.10	186.76	13	0.60
	IV	192.61	14	0.07	187.07	13	0.57
	V	192.56	14	0.07	186.95	13	0.58
	AVE	192.29	14	0.08	**186.99**	**13**	0.58
lse-91	I	23.46	23	0.32	25.11	18	5.49
	II	23.57	23	0.30	24.70	18	5.51
	III	23.49	23	0.29	24.81	18	5.45
	IV	23.51	23	0.30	24.95	18	5.40
	V	23.55	23	0.29	24.80	18	5.51
	AVE	**23.52**	23	0.3	24.87	**18**	5.47
kfu-93	I	36.12	25	0.41	35.26	20	8.49
	II	36.41	25	0.41	35.34	20	7.96
	III	36.45	25	0.40	35.19	20	7.92
	IV	36.54	25	0.40	35.30	20	8.02
	V	36.13	25	0.41	35.43	20	8.02
	AVE	36.33	25	0.41	**35.30**	**20**	8.08

Table 8: Comparative Computational Results of Selection Mechanisms

Dataset	**Run**	**Rank**			**Roulette Wheel**			**Tournament**		
		Penalty Cost	Total Timeslot	CPU Time (sec)	Penalty Cost	Total Timeslot	CPU Time (sec)	Penalty Cost	Total Timeslot	CPU Time (sec)
sta-83	I	169.14	13	239.90	169.33	13	15.66	169.52	13	15.41
	II	169.14	13	247.05	169.71	13	14.33	169.88	13	12.88
	III	169.57	13	263.52	169.26	13	14.56	169.14	13	15.53
	IV	169.29	13	254.21	169.99	13	13.46	168.94	13	16.47
	V	169.23	13	251.73	170.21	13	13.14	169.23	13	14.38
	AVE	**169.27**	**13**	251.28	169.70	**13**	14.23	169.34	**13**	14.93
lse-91	I	17.22	18	1213.08	16.40	18	73.42	16.90	18	60.29
	II	17.43	18	1127.63	17.03	18	77.04	16.39	18	60.53
	III	16.02	18	1046.33	15.71	18	74.10	16.49	18	59.52
	IV	17.62	18	1311.08	17.05	18	76.80	16.85	18	60.20
	V	17.60	18	1460.41	17.64	18	74.81	16.69	18	57.24
	AVE	17.18	**18**	1231.71	16.77	**18**	75.23	**16.66**	**18**	59.56
kfu-93	I	21.70	20	1706.44	21.08	20	77.78	19.83	20	71.66
	II	21.27	20	1311.65	20.57	20	77.57	19.70	20	73.60
	III	21.14	20	1494.26	20.89	20	78.52	20.66	20	76.55
	IV	20.39	20	1304.43	21.86	20	78.81	20.18	20	75.55
	V	21.07	20	1377.12	20.42	20	79.05	19.53	20	75.43
	AVE	21.11	**20**	1438.78	20.96	**20**	78.35	**19.98**	**20**	74.56

Table 9: Comparative Computational Results of Crossover Operators

Dataset	**Run**	**Modified One Point**			**Modified Two Point**			**Modified Position-Based**		
		Penalty Cost	Total Timeslot	CPU Time (sec)	Penalty Cost	Total Timeslot	CPU Time (sec)	Penalty Cost	Total Timeslot	CPU Time (sec)
sta-83	I	169.52	13	15.41	168.61	13	16.80	160.17	13	17.67
	II	169.88	13	12.88	169.89	13	18.29	160.03	13	17.01
	III	169.14	13	15.53	169.99	13	16.88	159.87	13	16.27
	IV	168.94	13	16.47	169.43	13	18.53	159.94	13	16.66
	V	169.23	13	14.38	169.44	13	19.11	159.52	13	18.90
	AVE	169.34	**13**	14.93	169.47	**13**	17.92	**159.91**	**13**	17.3
lse-91	I	16.90	18	60.29	16.64	18	57.79	15.56	19	69.47
	II	16.39	18	60.53	16.18	18	72.43	14.91	19	68.04
	III	16.49	18	59.52	15.89	18	68.74	14.70	19	68.58
	IV	16.85	18	60.20	15.72	18	71.81	14.07	19	68.16
	V	16.69	18	57.24	16.05	18	70.35	14.48	19	63.75
	AVE	16.66	**18**	59.56	16.10	**18**	68.22	**14.74**	19	67.60
kfu-93	I	19.83	20	71.66	19.42	20	88.38	17.67	21	76.24
	II	19.70	20	73.60	19.11	20	87.16	18.82	21	85.33
	III	20.66	20	76.55	20.33	20	87.02	17.13	21	85.75
	IV	20.18	20	75.55	18.81	20	87.03	16.40	22	85.34
	V	19.53	20	75.43	19.61	20	86.71	19.15	20	85.93
	AVE	19.98	**20**	74.56	19.46	**20**	87.26	**17.83**	21	83.72

Table 10: Comparative Computational Results of Replacement Strategies

Dataset	Run	Steady-State			Elitism			En-Bloc		
		Penalty Cost	Total Timeslot	CPU Time (sec)	Penalty Cost	Total Timeslot	CPU Time (sec)	Penalty Cost	Total Timeslot	CPU Time (sec)
sta-83	I	160.60	13	13.70	160.21	13	13.84	169.47	13	12.70
	II	160.27	13	13.06	160.41	13	11.94	169.38	13	12.80
	III	159.91	13	14.05	160.36	13	11.64	168.89	13	12.57
	IV	160.30	13	11.50	160.33	13	11.64	168.39	13	12.66
	V	159.65	13	12.94	160.66	13	11.95	168.06	13	12.60
	AVE	**160.15**	**13**	13.05	160.39	**13**	12.20	168.84	**13**	12.67
lse-91	I	15.55	18	55.24	14.89	19	53.63	18.87	25	43.31
	II	15.89	18	58.06	15.13	19	55.12	19.49	25	42.85
	III	15.92	18	56.41	15.93	18	55.00	19.06	25	43.13
	IV	16.42	18	57.13	14.87	19	56.68	19.23	25	42.63
	V	16.05	18	56.69	17.02	18	58.53	18.34	25	42.66
	AVE	15.97	**18**	56.71	**15.57**	19	55.79	19.00	25	42.92
kfu-93	I	19.02	20	71.41	18.17	20	65.06	19.63	20	70.38
	II	18.21	20	71.69	19.69	20	67.14	19.90	20	70.02
	III	18.84	20	68.09	19.78	20	65.29	17.65	21	69.44
	IV	19.78	20	71.95	19.74	20	69.96	17.67	21	68.74
	V	20.78	20	71.02	18.52	20	72.26	17.57	21	72.45
	AVE	19.33	**20**	70.83	19.18	**20**	67.94	**18.48**	21	70.21

Table 11: Comparative Computational Results of Mutation Operators

Dataset	Run	Inverse			Swap			Scramble			Insertion		
		Penalty Cost	Total Timeslot	CPU Time (sec)	Penalty Cost	Total Timeslot	CPU Time (sec)	Penalty Cost	Total Timeslot	CPU Time (sec)	Penalty Cost	Total Timeslot	CPU Time (sec)
sta-83	I	168.61	13	16.80	169.19	13	15.11	160.60	13	12.94	159.97	13	13.68
	II	169.89	13	18.29	170.62	13	15.01	160.27	13	11.50	160.66	13	12.10
	III	169.99	13	16.88	170.51	13	13.19	160.30	13	13.06	160.22	13	13.85
	IV	169.43	13	18.53	169.39	13	12.68	159.91	13	14.05	161.03	13	11.74
	V	169.44	13	19.11	168.65	13	15.73	159.65	13	13.70	160.20	13	13.62
	AVE	169.47	**13**	17.92	169.67	**13**	14.34	**160.15**	**13**	13.05	160.42	**13**	13.00
lse-91	I	16.64	18	57.79	15.12	20	52.13	15.55	18	55.24	16.26	18	56.97
	II	16.18	18	72.43	13.83	20	54.66	15.89	18	58.06	17.02	18	58.83
	III	15.89	18	68.74	14.67	19	55.06	15.92	18	56.41	18.52	19	58.36
	IV	15.72	18	71.81	16.59	19	58.50	16.42	18	57.13	15.46	19	59.57
	V	16.05	18	70.35	16.96	18	59.29	16.05	18	56.69	14.56	19	59.43
	AVE	16.10	**18**	68.22	**15.43**	19	55.93	15.97	**18**	56.71	16.36	19	58.63
kfu-93	I	19.42	20	88.38	16.58	22	67.54	19.02	20	71.41	17.63	21	76.09
	II	19.11	20	87.16	19.92	20	72.20	18.21	20	71.69	19.66	21	74.37
	III	20.33	20	87.02	16.68	22	68.91	18.84	20	68.09	21.36	21	71.18
	IV	18.81	20	87.03	16.98	22	65.42	19.78	20	71.95	17.90	21	72.37
	V	19.61	20	86.71	18.97	21	71.88	20.78	20	71.02	18.64	20	72.14
	AVE	19.46	**20**	87.26	**17.83**	21	69.19	19.33	**20**	70.83	19.04	21	73.23